

# Requirements Specification for

## Context Aware Mobile Application Testing Framework (CAMATF)

September 20, 2016  
Version 0.1

### **Prepared By**

Mutasim Fuad Ansari  
BSSE0520

### **Supervised By**

Amit Seal Ami  
Lecturer  
Institute Of Information Technology  
University Of Dhaka

## LETTER OF TRANSMITTAL

20<sup>th</sup> September, 2016

Dr. Mohammad Shoyaib

Associate Professor

Institute of Information Technology

University of Dhaka

Sir,

We have prepared the report on Software Requirements Specification of 'Context Aware Mobile App Testing Framework' for your approval. This report details the requirements we gathered for the project.

The primary purpose of this report is to summarize our findings from the work that we completed as our Software Requirements Specification and Analysis course project. This report includes the details of each step we followed to collect the requirements.

Sincerely Yours,

Mutasim Fuad Ansari

(BSSE-0520)

## Table of Contents

Chapter 1: Introduction .....	1
1.1 Purpose .....	1
1.2 Scope .....	1
1.3 Overview .....	1
Chapter 2: Inception .....	2
2.1 Creating a preliminary scenario .....	2
2.2 Identifying Stakeholders .....	2
2.3 Asking First Questions.....	2
2.4 Recognizing multiple view points .....	3
Chapter 3: Elicitation.....	5
3.1 Eliciting Requirements .....	5
3.2 Quality Function Deployment (QFD).....	5
Chapter 4: Scenario Based Modeling.....	6
Scenario Based Modelling.....	6
Use Case.....	6
Activity Diagram.....	6
Swimlane Diagram .....	6
Chapter 5: Databased Modelling .....	20
5.1 Data Object Identification.....	20
5.2 Data Object and Attributes .....	21
5.3 Entity Relationship Diagram.....	22
5.4 Schema.....	24
Chapter 6: Class Based Modeling.....	26
6.1 General Classification.....	26
6.2 Selection Characteristic.....	27
6.3 Specifying Attributes.....	28
6.4 Defining Operations .....	29
6.5 Class Relation Diagram.....	29
6.6 Class-Responsibility-Collaborator (CRC) Model .....	31
Chapter 7: Behavioral Modeling .....	32
7.1 Identifying Events.....	32
7.2 State Diagram.....	33

7.3 Sequence Diagram .....	36
Chapter 8: Conclusion .....	40
Chapter 9: References.....	41

# Chapter 1: Introduction

## 1.1 Purpose

The purpose of this document is to present a detailed description of a web based online mobile application testing tool named “Context Aware Mobile Application Testing Framework”. It will explain the purpose and features of the testing framework, the interfaces of the framework, what the procedure of framework will follow. This document is intended for both the stakeholders and the developers of the game.

## 1.2 Scope

This documents covers the eliciting requirements, QFD of the project, scenario based modeling, databased modeling, class based modeling and behavioral modeling of the project

## 1.3 Overview

The second chapter, Inception, of this document describes how the functional requirements are gained by talking with stakeholders, meeting etc. The third chapter, Elicitation elicits the requirements gained via Inception. The next chapters, Scenario Based Modeling, Data Based Modeling, Class Based Modeling, Data Flow Diagram, and Behavioral Model are written primarily for the developers and describes in technical terms the details of the functionality of the product.

## Chapter 2: Inception

When a business need is identified or new potential market product or service is discovered, a software project begins. Stakeholders from the business community define a business case for the idea, try to identify breadth and depth of the market, do a rough feasibility analysis and identify a working description. At inception, we established a basic understanding of the problem.

### 2.1 Creating a preliminary scenario

Statistics show that mobile apps are not being tested properly. According to survey data, nearly 50% of consumers will delete a mobile app if they find a single bug [1]. Study researchers found three-quarters (77%) of mobile users are concerned about app performance before they buy and that half (51%) of app developers say they 'don't have time' to properly test apps before release [1]. The problem is that testing mobile applications is no easy task and generally requires a great deal of manual and exploratory testing. So if the testing process could be automated both time and cost could be saved for the developers. The purpose of this project is to build a framework that will automatically test the mobile apps in different contexts. CAMATF is a web based mobile application testing tool where mobile apps developer/tester can easily test their mobile application in different contexts such as display size, RAM, CPU etc.

### 2.2 Identifying Stakeholders

Stakeholder refers to any person or group who will be affected by the system directly or indirectly. Stakeholders include end-users who interact with the system and everyone else in an organization that may be affected by its installation.

We identified the stakeholders for our mobile apps testing framework as followings:

- Mobile apps developers
- Mobile apps testers

### 2.3 Asking First Questions

We set our first set of context-free questions focusing on the stakeholders, overall project goals and benefits. The questions are mentioned below. These questions helped us to identify all stakeholders, measurable benefit of the successful implementation and possible alternatives to custom game development. Next set of questions helped us to gain a better understanding of problem and allows the gamer to voice their perception

about the solution. The final set of question focused on the effectiveness of the communication activity itself.

1. What will be tested?
2. On which criteria the apps will be tested?
3. What will be the contexts?
4. How the test case will be generated?
5. Will the user be free to select their contexts?
6. What should be the business value?
7. How should be the UI?
8. Do the user want to store their test results?
9. What kind of mobile apps will be tested?
10. Will the user be free the value of the selected contexts?

## 2.4 Recognizing multiple view points

After the discussion with the stakeholders we collected multiple viewpoints.

1. Source Code/APK will be tested in CAMATF
2. User must have an account to test their apps
3. Contexts will be display size, CPU, RAM, network delay, packet loss, system image
4. The value of the contexts will be given from where the user will choose her desired value of that context
5. User can see their previous tests
6. Only android apps will be tested
7. User will give the test cases

After consulting with my supervisor and the stakeholders the second iteration of the story is following:

*CAMATF will test different kinds of mobile applications against various contexts. Users do not have keep device of various configuration to test their app. This framework will give that platform to the users. It will test android apps. To test application users have to give the source Code/APK and test cases as input. The given input will be tested against the following contexts: display size, RAM, CPU, network delay, packet loss, GPRS and system image. Every context category will have some given value from where the user will choose their desired value.*

*Users need to have account for testing their application in this framework. User will need a valid email, username and password to create an account. Every test that has been run successfully form an account will be stored so that the user of that account can compare their next testing with the previous one.*

*This framework will have three different modules which are web, distribution and testing module. User will give his inputs through the web module. All their inputs will be then sent to the distribution module where Node Manager will create distributed system and. Then node manager will send user's information, apps and test cases to emulator manager.*

*Node manager may have many nodes under it. Every Node has an emulator manager. Emulator manager manage the testing module. An emulator manager may have many virtual device under testing. From the user's given context emulator manager creates virtual device's configuration file. After creating configuration file virtual device will be created. After creating the virtual device emulator manager will test the device according to the user's given test-case. During testing emulator manager will monitor every virtual device under testing. Every report in the testing phase will be record in a log file. If any of the devices stop working before the testing finishes emulator manager will re-instantiate the device again and run the test again. After a successful testing emulator manager will gather all the result and send it to the node manager. Node manager will send the results, found from different nodes to the web module where the results will be stored and displayed.*














# Chapter 3: Elicitation

## 3.1 Eliciting Requirements

Unlike inception where Q&A (Question and Answer) approach is used, elicitation makes use of a requirements elicitation format that combines the elements of problem solving, elaboration, negotiation, and specification. It requires the cooperation of a group of end-users and developers to elicit requirements.

## 3.2 Quality Function Deployment (QFD)

Quality Function Deployment (QFD) is a technique that translates the needs of the customer into technical requirements for software. It concentrates on maximizing customer satisfaction from the software engineering process. With respect to our project the following requirements are identified by a QFD.

- Normal Requirements
  -  *Account creation*
  -  *Upload android Source Code/ APK to test*
  -  *Upload test case*
  -  *Select contexts*
  -  *Viewing previous test results*
- Expected Requirements
  -  *Test result will be visualized*
  -  *Proper authentication*
  -  *User can select the values of*
  -  *Better UI design*
  -  *User has a completely developed app*
- Exciting Requirements
  -  *Caching the virtual device*

# Chapter 4: Scenario Based Modeling

In this modelling the system is described from the user's point of view using a scenario based approach. Basically scenario based modelling evolves basic use case, use case diagrams (activity and swimlane diagram).

## Scenario Based Modelling

In this modelling the system is described from the user's point of view using a scenario based approach. Basically scenario based modelling evolves basic usecase, use case diagrams (activity and swimlane diagram).

## Use Case

Use case describes a specific usage scenario in straight forward language from the point of view of a defined actor.

## Activity Diagram

Activity diagram supplements the use case by providing a graphical presentation of the flow of interaction within a scenario. Similar to the flow chart the activity diagram to simply identifies the process, diamond for a decision.

## Swimlane Diagram

Swimlane diagram is a variation of activity diagram which represents the flow of activities described by the usecase and at the same time indicate which actor is responsible for the actions described by the activity rectangles.

In figure 4.1 I have shown the figure of usecase CAMATF. This is usecase level -0

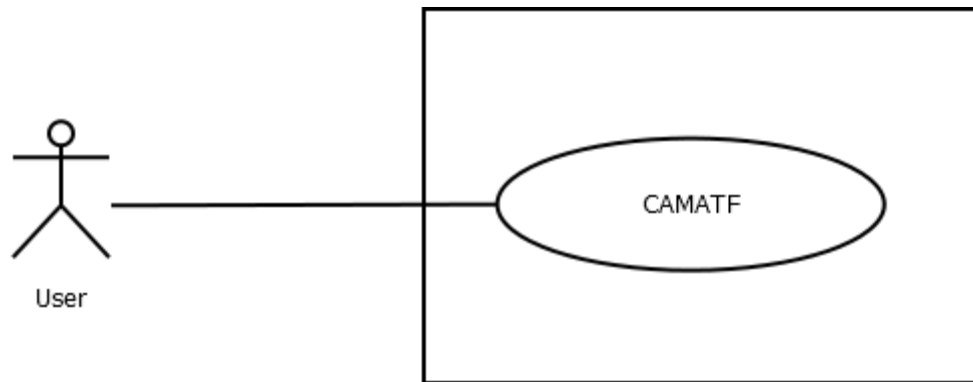


Figure 4.1: CAMATF Use case level – 0

Use Case Name	CAMATF
Primary Actor	User
Secondary Actor	System
Goal in context	To test android applications in different contexts.
Preconditions	User must have account in CAMATF.
Actions	This is very naïve level.
Exception	No exception

Use case level – 0 is very initial level. At this level activity diagram and corresponding swim lane diagram is not necessary.

In figure 4.2 I have shown the figure of usecase of Authentication, Login, Registration, Testing, and View Previous Result. This is usecase level - 1

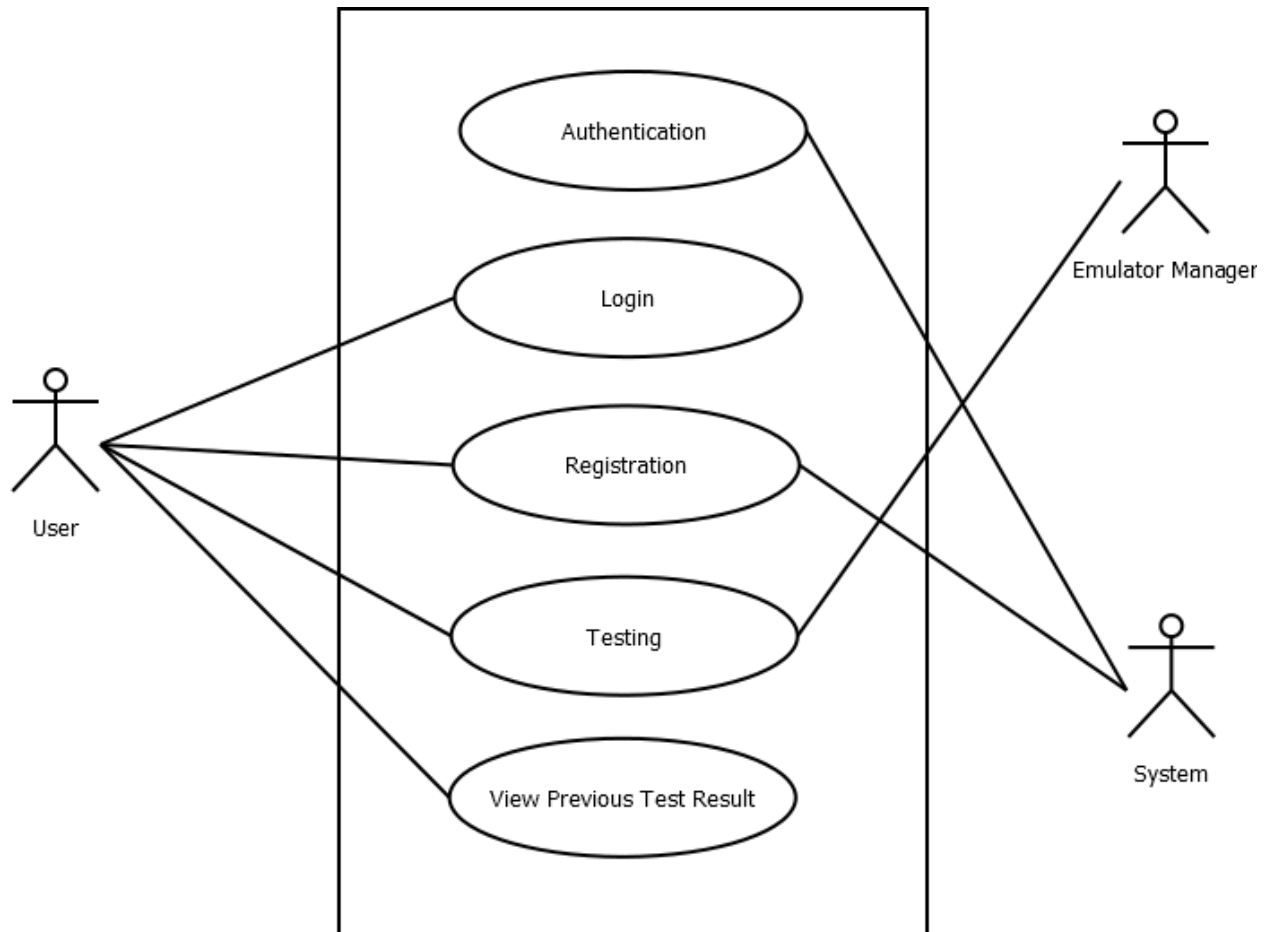


Figure 4.2: use case level – 1

Use case Name	Login
Primary Actor	User
Secondary Actor	System
Goal in context	To log into account
Preconditions	Must have an Account
Trigger	User is ready to sign into her account
Actions	<ol style="list-style-type: none"> <li>1. User fills up the necessary fields</li> <li>2. Click the login button</li> <li>3. If verified logged in</li> </ol>
Exception	No exception

Use case Name	Authentication
Primary Actor	System
Secondary Actor	User
Goal in context	To verify the users account
Preconditions	Must have an Account
Trigger	User is ready to sign into his account
Actions	<ol style="list-style-type: none"> <li>1. User fills up the necessary fields</li> <li>2. Click the login button</li> <li>3. If verified logged in</li> </ol>
Exception	No exception

Use case Name	Registration
Primary Actor	System, User
Secondary Actor	No secondary actor
Goal in context	To create account
Preconditions	Must have an valid email address, username and password
Trigger	User is ready to create her account
Actions	<ol style="list-style-type: none"> <li>1. User fills up the necessary fields</li> <li>2. Click the confirm button</li> <li>3. If verified account created</li> </ol>
Exception	No exception

Use case Name	Testing
Primary Actor	Emulator Manager
Secondary Actor	User
Goal in context	To test the application
Preconditions	Must have account
Trigger	User is ready to test her application
Actions	<ol style="list-style-type: none"> <li>1. User fill up the necessary contexts fields</li> <li>2. Upload the APK\source code</li> <li>3. Click the run test button</li> </ol>
Exception	No exception

Use case Name	View Previous Test result
Primary Actor	User
Secondary Actor	System
Goal in context	To view any previous test result if available
Preconditions	Must have any previous test results
Trigger	User is ready to see her previous test results
Actions	<ol style="list-style-type: none"> <li>1. User will click the previous result button</li> <li>2. User will choose the desired result from a list</li> <li>3. Next user will click the view button</li> </ol>
Exception	No exception

In figure 4.3 I have shown the figure of activity diagram of usecase - Authentication, Login, Registration, Testing, and View Previous Result.

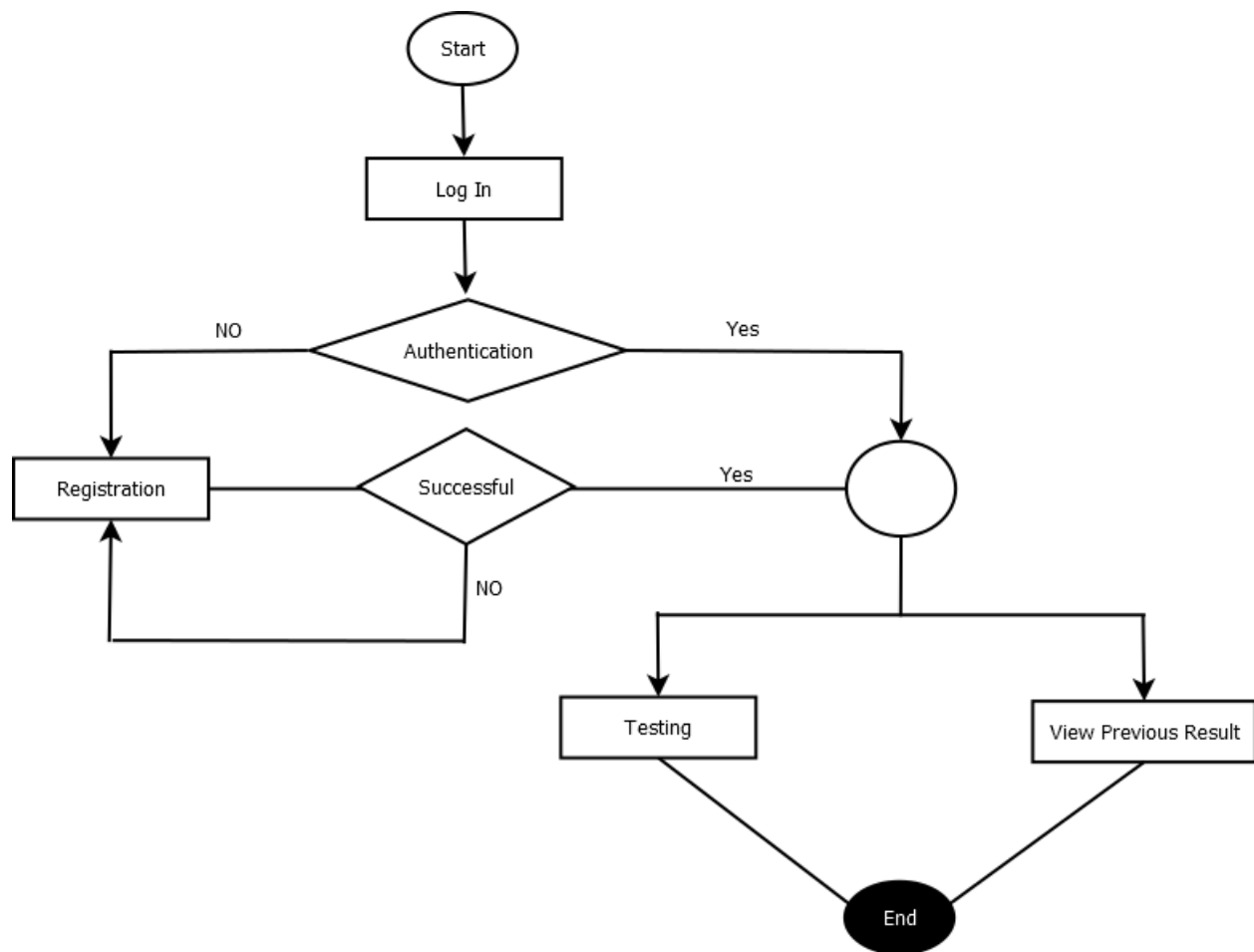


Figure 4.3: Activity Diagram of Use case level – 1

In figure 4.4 I have shown the figure of swimlane diagram of use case - Authentication, Login, Registration, Testing, and View Previous Result. This is usecase level - 1

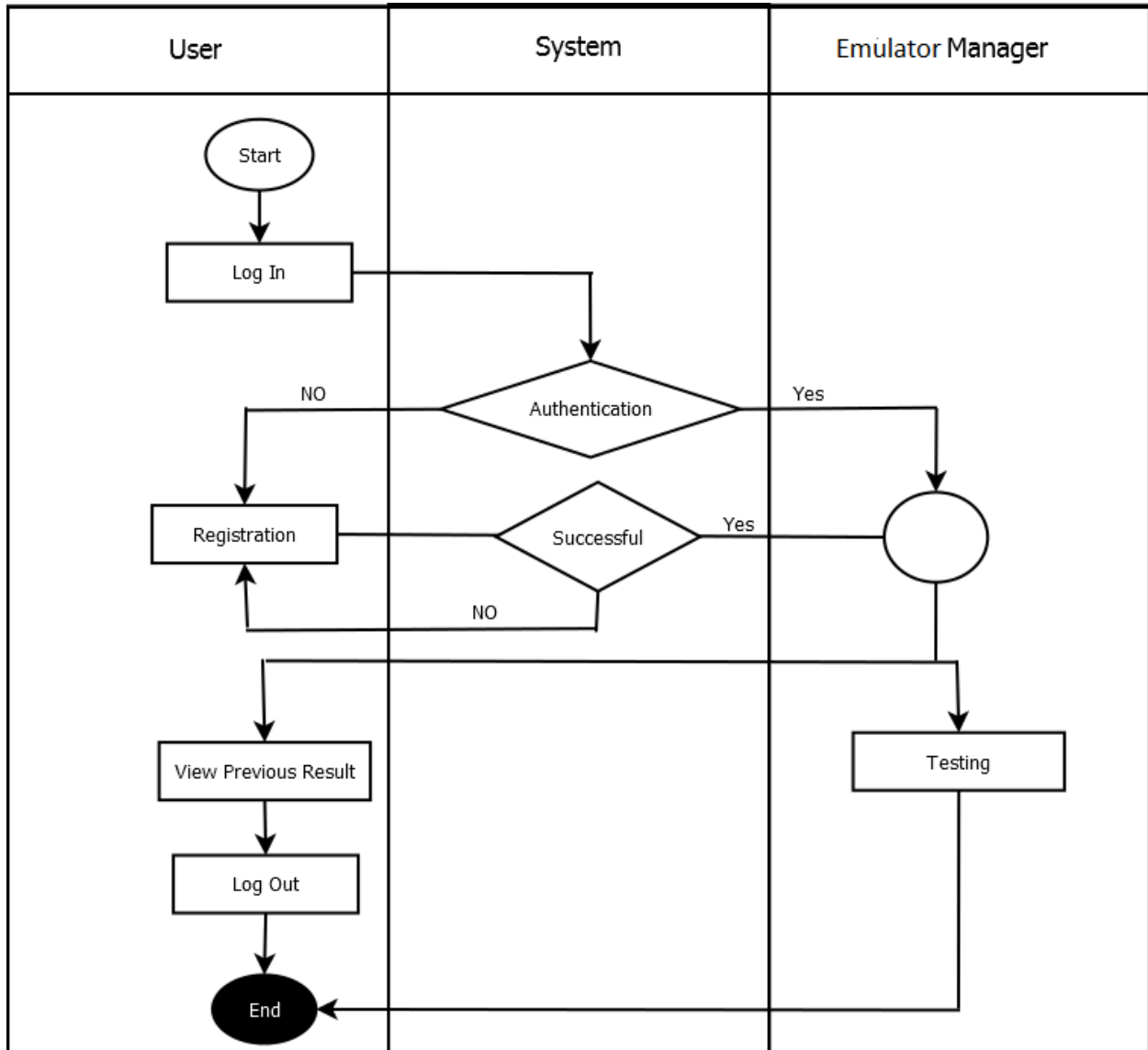


Figure 4.4: Swim Lane Diagram of Use case level – 1

In figure 4.5 I have shown the figure of usecase – Emulator Creation & Management, Result Analysis, Caching. This is usecase level – 1.4

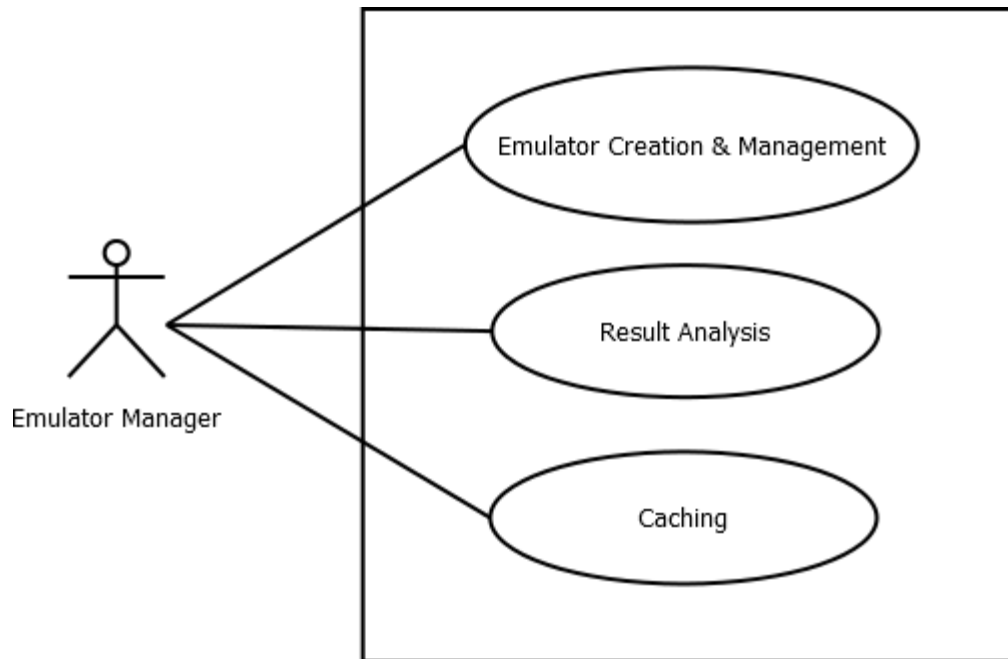


Figure 4.5: Use case level – 1.4

Use case Name	Emulator Creation & Management
Primary Actor	Emulator Manager
Secondary Actor	System
Goal in context	<ol style="list-style-type: none"> <li>1. Create virtual device configuration file</li> <li>2. Create virtual device</li> <li>3. Run test in the devices</li> <li>4. Monitor device</li> </ol>
Preconditions	Must have an account and deliver required contexts.
Trigger	User clicks the run test button
Actions	<ol style="list-style-type: none"> <li>1. User fills up the necessary fields of contexts</li> <li>2. Click the run test button</li> </ol>
Exception	No exception



Use case Name	Result Analysis
Primary Actor	Emulator Manager
Secondary Actor	System, User
Goal in context	<ol style="list-style-type: none"> <li>1. To gather test result</li> <li>2. Visualize test result</li> </ol>
Preconditions	Must have completed tests according to the test cases successfully in the virtual device
Trigger	Virtual device have been created and ready for testing
Actions	<ol style="list-style-type: none"> <li>1. User clicks the run test button</li> </ol>
Exception	No exception

Use case Name	Caching
Primary Actor	Emulator Manager
Secondary Actor	System, User
Goal in context	<ol style="list-style-type: none"> <li>3. To gather test result</li> <li>4. Visualize test result</li> </ol>
Preconditions	Must have completed tests according to the test cases successfully in the virtual device
Trigger	Virtual device have been created and ready for testing
Actions	<ol style="list-style-type: none"> <li>2. User clicks the run test button</li> </ol>
Exception	No exception

In figure 4.6 I have shown the figure of activity diagram of usecase - Emulator Creation & Management, Result Analysis, Caching. This is usecase level – 1.4

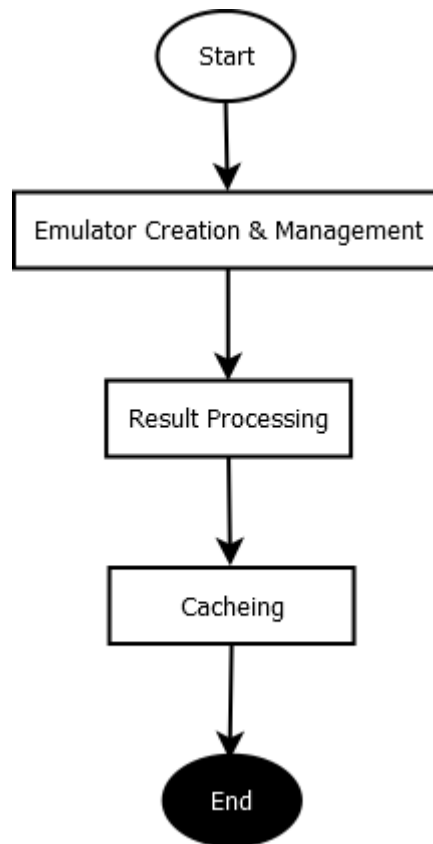


Figure 4.6: Activity Diagram of use case level – 1.4

In figure 4.7 I have shown the figure of swimlane diagram of usecase - Emulator Creation & Management, Result Analysis, Caching. This is usecase level – 1.4

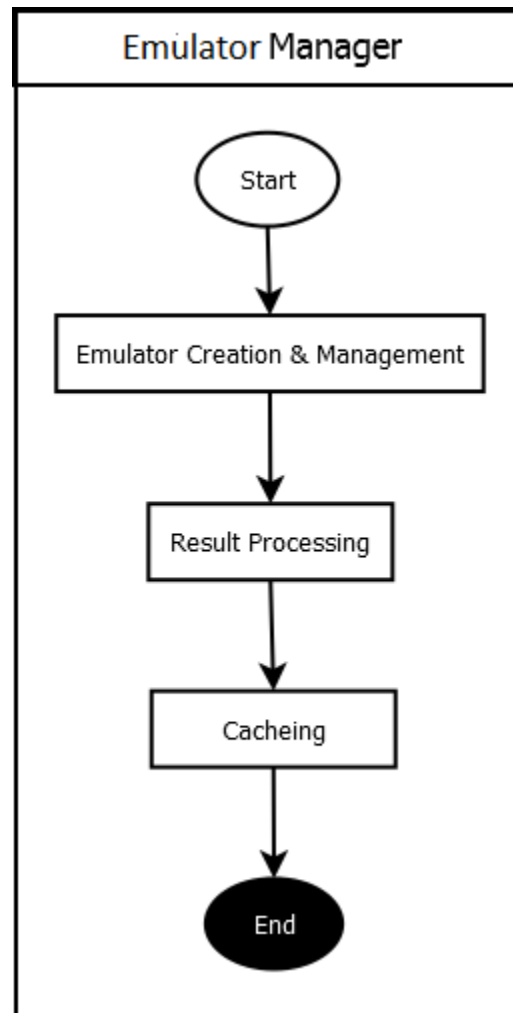


Figure 4.7: Swimlane diagram of use case level – 1.4

In figure 4.8 I have shown the figure of usecase diagram of usecase – Configuration File Creation, Monitor Emulator and Emulator Re-instantiate. This is usecase level – 1.4.1

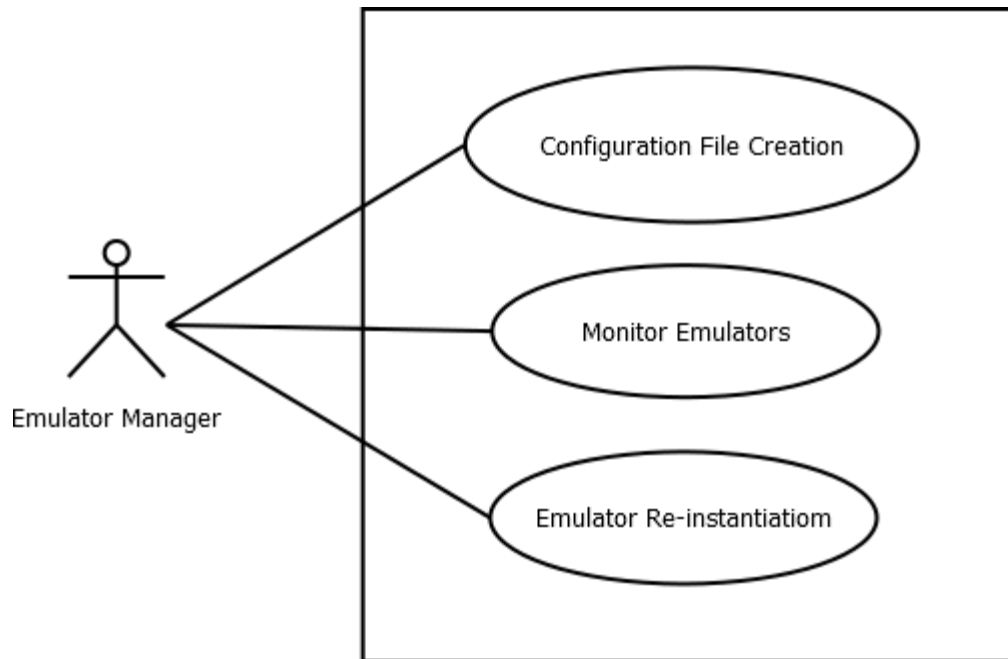


Figure 4.8: use case level – 1.4.1

Use case Name	Configuration File Creation
Primary Actor	Emulator Manager
Secondary Actor	System, User
Goal in context	To create configuration file from user's given contexts
Preconditions	Must have to receive contexts
Trigger	Node manager has send the contexts to the emulator manager
Actions	
Exception	No exception

Use case Name	Monitor Emulators
Primary Actor	Emulator Manager
Secondary Actor	System
Goal in context	To monitor every virtual device if they are working properly or not.
Preconditions	Must have been created the virtual devices
Trigger	The virtual device has been created and the test are running
Actions	
Exception	No exception

Use case Name	Emulator Re-instantiation
Primary Actor	Emulator Manager
Secondary Actor	System
Goal in context	To re-instantiate every emulator if any virtual device stops working before the test finishes.
Preconditions	Test has been initiated and error has occurred.
Trigger	Virtual device has stopped working
Actions	
Exception	No exception

In figure 4.9 I have shown the figure of activity diagram of usecase – Configuration File Creation, Monitor Emulator and Emulator Re-instantiate. This is usecase level – 1.4.4

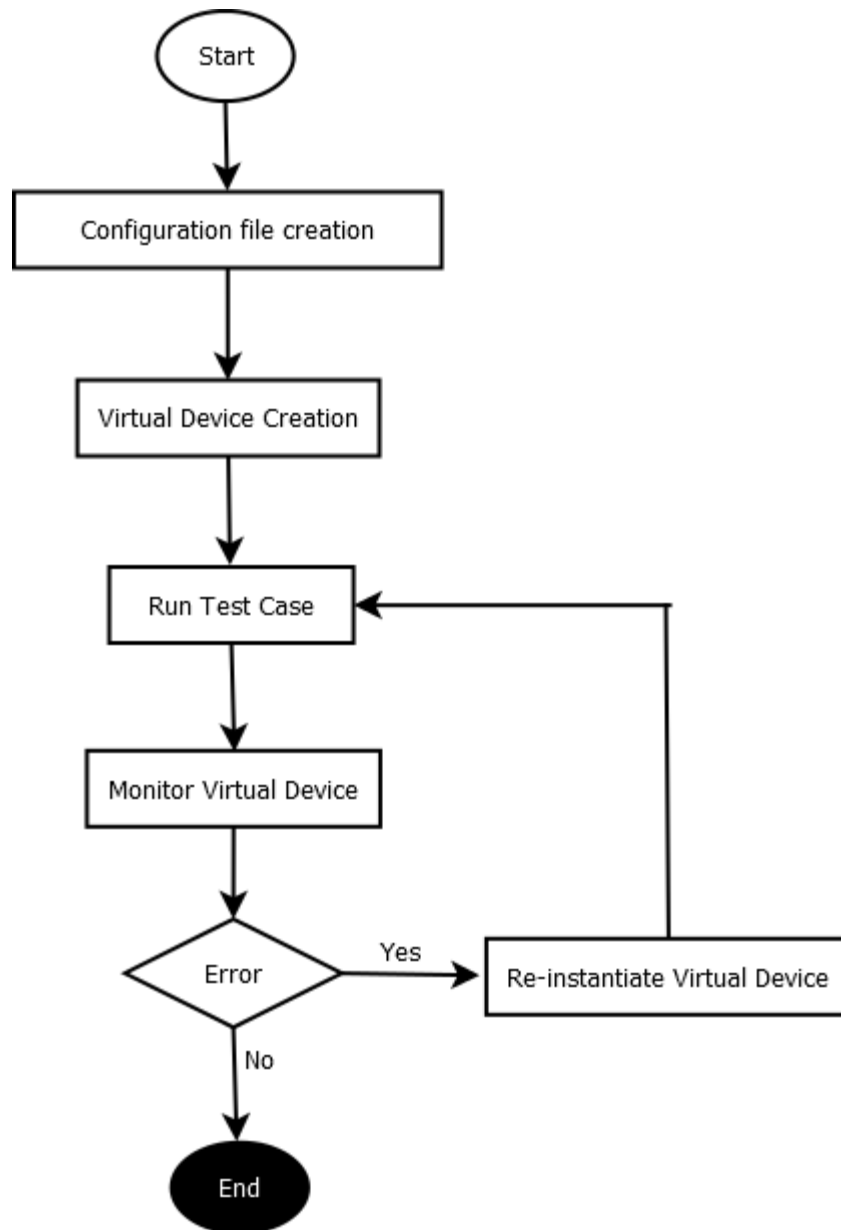


Figure 4.9: Activity diagram of use case level - 1.4.1

In figure 4.10 I have shown the figure of swimlane diagram of usecase – Configuration File Creation, Monitor Emulator and Emulator Re-instantiate. This is usecase level – 1.4.1

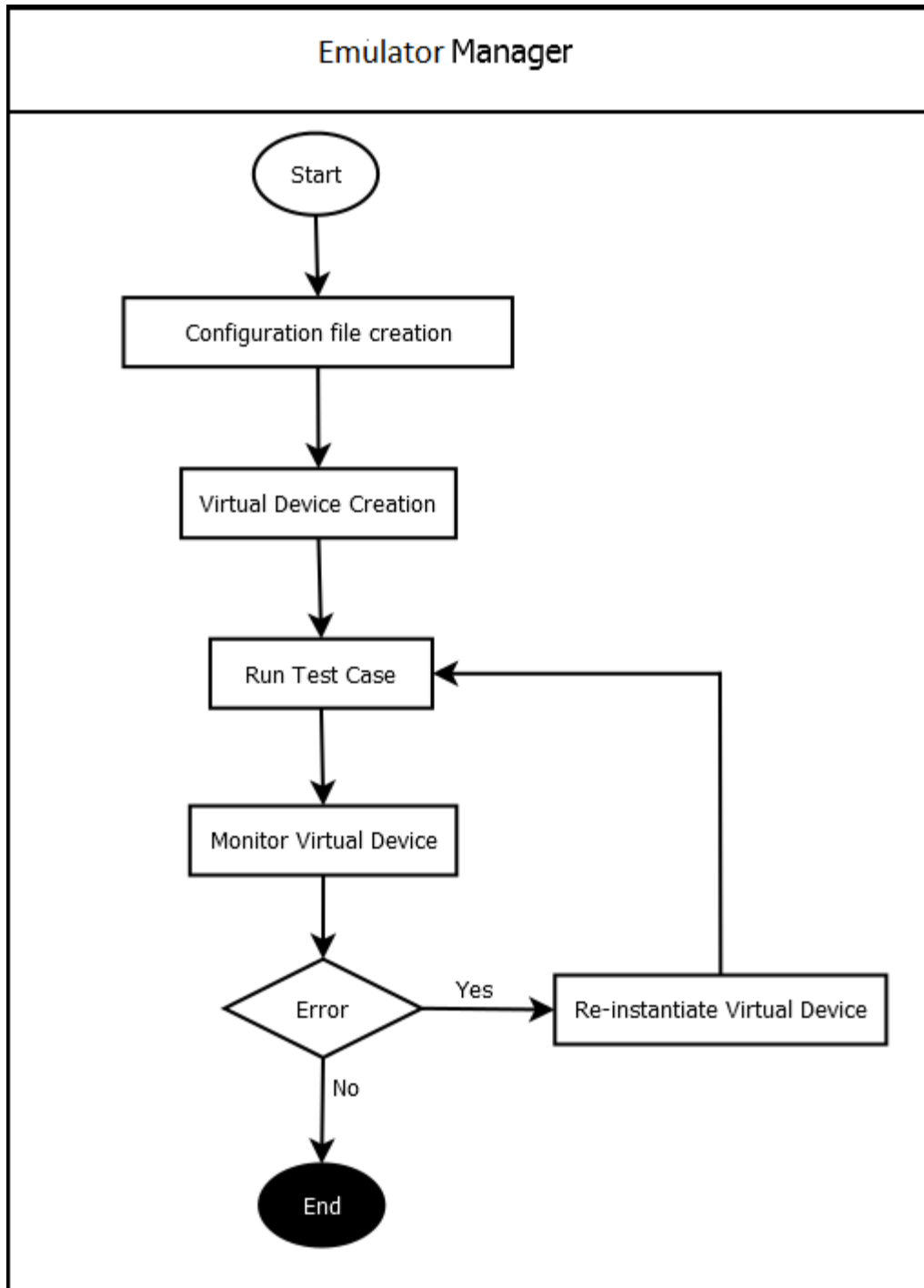


Figure 4.10: Swim Lane Diagram of use case level – 1.4.1

## Chapter 5: Databased Modelling

If software requirements include the need to create, extend, or interface with a database or if complex data structures must be constructed and manipulated, the software team may choose to create a data model as part of overall requirements modeling.

### 5.1 Data Object Identification

In Table 5.1 I have shown the Table of data object identification

No	Noun	General Classification	Remarks
1	CAMATF	Null	Solution Space
2	Mobile application	Thing	Problem Space
3	Contexts	Thing	Solution Space
4	User	Role	Problem Space
5	Configuration	Thing	Solution Space
6	Registration	Occurrences	Solution Space
7	Code/APK	Thing	Solution Space
8	Instantiation	Occurrence	Solution Space
9	Display size	Thing	Solution Space
10	RAM	Thing	Solution Space
11	CPU	Thing	Solution Space
12	Network delay	Thing	Solution Space
13	Packet loss	Thing	Solution Space
14	System Image	Thing	Solution Space
15	Context category	Thing	Solution Space
16	Email	Thing	Solution Space
17	Web module	Null	Solution Space
18	Monitor	Occurrence	Solution Space
19	Distribution module	Null	Problem Space
20	Node manager	Role	Solution Space
21	Test cases	Thing	Solution Space
22	Emulator management	Role	Solution Space
23	Node	Place	Solution Space
24	Emulator instantiation	Occurrence	Solution Space
25	User 's information	Thing	Problem Space
26	Emulator manager	Role	Solution Space
27	Virtual device	Thing	Solution Space
28	Configuration file	Occurrence	Solution Space
29	Results	Thing	Solution Space
30	External context	Thing	Solution Space



31	Internal context	Thing	Solution Space
32	Log	Thing	Solution Space
33	Email	Thing	Solution Space
34	Web module	Null	Solution Space
35	Monitor	Occurrence	Solution Space
36	Email	Thing	Solution Space
37	Node	Place	Solution Space

Table 5.1: Table of data object identification

## 5.2 Data Object and Attributes

- User
  - *User\_id*
  - *Username*
  - *Email*
  - *Password*
- Result
  - *Result\_id*
  - *numberOfTest*
  - *numberOfPassedTest*
  - *numberOfFailedTest*
  - *failedTestCase*
  - *time*
  - *screenshot*
  - *test\_id*
  - *device\_id*
- ExternalContext
  - *Orientation*
  - *GPRS*
  - *NetworkSwitch*
  - *networkDelay*
  - *packetloss*
  - *device\_id*
- InternalContext

- *RAM*
- *CPU*
- *DisplaySize*
- *systemImage*
- *device\_id*
- TestHistory
  - *Test\_Id*
  - *User\_id*
  - *Result\_id*
  - *Number\_of\_test*
  - *versionOfTest*
  - *device\_id*
- Device
  - *device\_id*
  - *user\_id*
  - *testCase*

## 5.3 Entity Relationship Diagram

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

In Figure 5.1 I have shown the ER Diagram of CAMATF

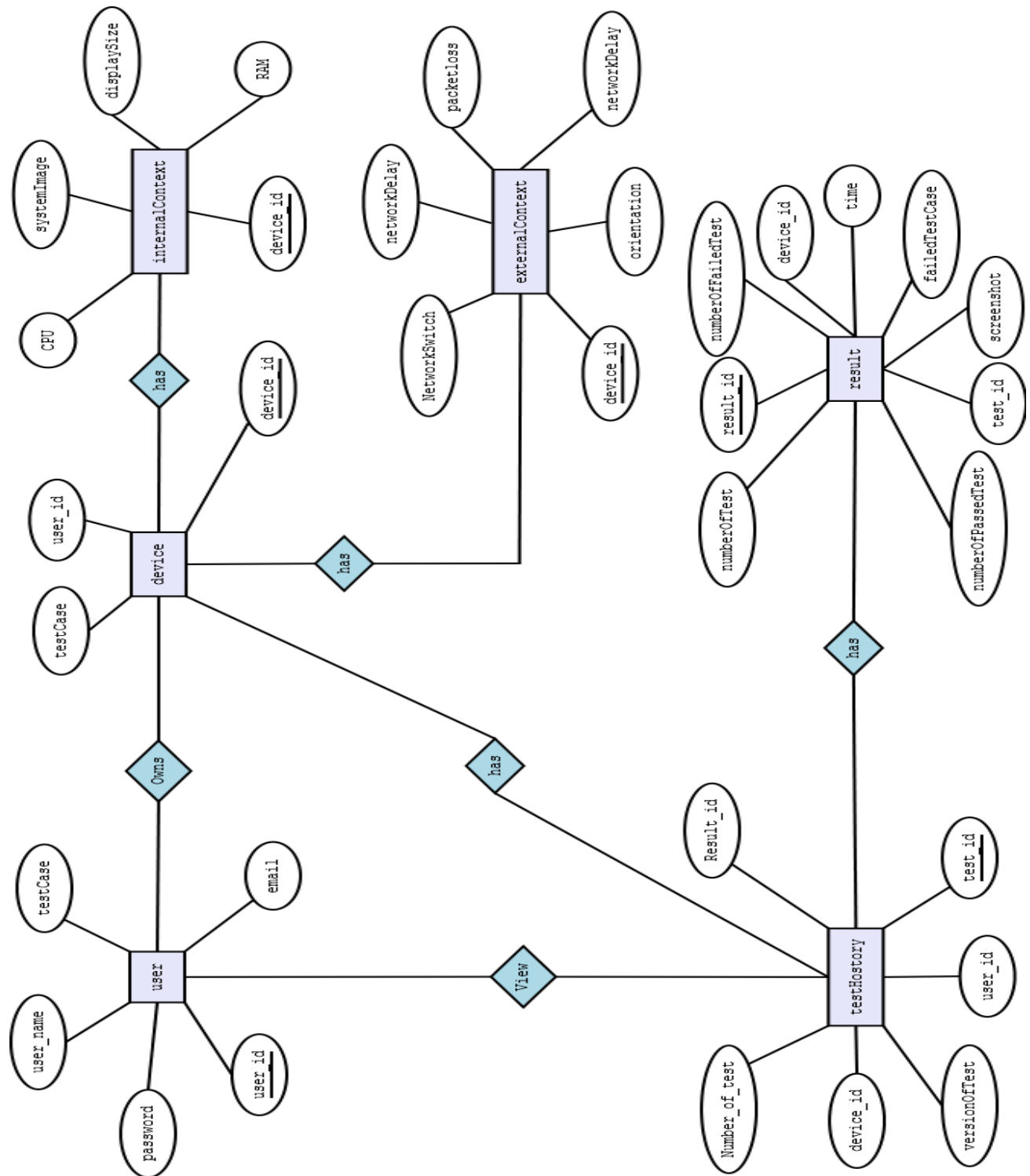


Figure 5.1: ER Diagram of CAMATF

## 5.4 Schema

Table Name: User	
Attribute	Data Type
<i>user_id</i>	Number
<i>username</i>	VARCHAR(50)
<i>email</i>	VARCHAR(50)
<i>password</i>	VARCHAR(50)

Table Name: Result	
Attribute	Data Type
<i>result_id</i>	Number
<i>numberOfTest</i>	Number
<i>numberOfPassedTest</i>	Number
<i>numberOfFailedTest</i>	Number
<i>failedTestCase</i>	STRING
<i>time</i>	DATE
<i>screenshot</i>	STRING
<i>test_id</i>	Number
<i>device_id</i>	Number

Table Name: ExternalContext	
Attribute	Data Type
<i>orientation</i>	VARCHAR(10)
<i>GPRS</i>	VARCHAR(10)
<i>networkSwitch</i>	VARCHAR(10)
<i>networkDelay</i>	VARCHAR(10)
<i>packetloss</i>	VARCHAR(10)
<i>device_id</i>	NUMBER

Table Name: InternalContext	
Attribute	Data Type
<i>test_id</i>	NUMBER
<i>user_id</i>	NUMBER
<i>result_id</i>	NUMBER
<i>number_of_test</i>	NUMBER
<i>versionOfTest</i>	VARCHAR(10)
<i>numberOfTest</i>	NUMBER

Table Name: TestHistory	
Attribute	Data Type
<i>orientation</i>	VARCHAR(10)
<i>GPRS</i>	VARCHAR(10)
<i>networkSwitch</i>	VARCHAR(10)
<i>networkDelay</i>	VARCHAR(10)
<i>packetloss</i>	VARCHAR(10)
<i>device_id</i>	NUMBER

Table Name: Device	
Attribute	Data Type
<i>device_id</i>	NUMBER
<i>user_id</i>	NUMBER
<i>testCase</i>	TEXT

# Chapter 6: Class Based Modeling

Class-based modeling represents the objects that the system will manipulate, the operations (also called methods or services) that will be applied to the objects to effect the manipulation, relationships (some hierarchical) between the objects, and the collaborations that occur between the classes that are defined. The elements of a class-based model include classes and objects, attributes, operations, class responsibility-collaborator (CRC) models.

## 6.1 General Classification

Classes are determined by underlining each noun or noun phrase and entering it into a simple table. Synonyms should be noted. After identifying noun or noun phrases analysis classes should manifest themselves in one of the following ways:

- i. External entities*
- ii. Things*
- iii. Occurrences*
- iv. Roles*
- v. Organizational units*
- vi. Places*
- vii. Structures*

In Table 6.1 I have shown the table of general classification of CAMATF

No	Noun	General Classification	Remarks
1	CAMATF	Null	No
2	Mobile application	Thing	No
3	Contexts	Thing	Potential Class
4	User	Role	Potential Class
5	Configuration	Thing	Potential Class
6	Registration	Occurrences	Potential Class
7	Code/APK	Thing	No
8	Instantiation	Occurrence	Potential Class
9	Display size	Thing	No
10	RAM	Thing	No
11	CPU	Thing	No
12	Network delay	Thing	No
13	Packet loss	Thing	No
14	System Image	Thing	No
15	Context category	Thing	No

16	Email	Thing	No
17	Web module	Null	No
18	Monitor	Occurrence	Potential Class
19	Distribution module	Null	No
20	Node manager	Role	No
21	Test cases	Thing	No
22	Emulator management	Role	Potential Class
23	Node	Place	No
24	Emulator instantiation	Occurrence	Potential Class
25	User 's information	Thing	No
26	Emulator manager	Role	Potential Class
27	Virtual device	Thing	No
28	Configuration file	Occurrence	Potential Class
29	Results	Thing	Potential Class
30	System	Role	Potential Class

Table 6.1: Table of general classification

## 6.2 Selection Characteristic

Coad and Yourdon suggest six selection characteristics that should be used as you consider each potential class for inclusion in the analysis model:

1. *Retained Information*
2. *Needed Services*
3. *Multiple Attributes*
4. *Common attributes*
5. *Common operations*
6. *Essential requirements*

In Table 6.2 I have shown the table of general selection characteristic of CAMATF

No	Noun	Selection Characteristic	Remarks
1	Contexts	4	Rejected
2	User	All	Accepted
3	Instantiation	2	Rejected
4	Monitor	2	Rejected
5	Emulator manager	All	Accepted
6	ContextConfiguration	1,2,6	Accepted
7	Results	1,2	Rejected
8	System	1,2,3,6	Accepted

Table 6.2: Table of selection characteristic

## 6.3 Specifying Attributes

Attributes describe a class that has been selected for inclusion in the requirements model. In essence, it is the attributes that define the class—that clarify what is meant by the class in the context of the problem space

In Table 6.3 I have shown the table of specifying attribute of CAMATF

Noun	Attributes
User	+numberOfPreviousTest +previousTests +result
Configuration	+display size +RAM +CPU +networkDelay +packetLoss +systemImage +configurationFile +virtualDevice
EmulatorManager	+virtualDevice +testCase +loggingFile +Result +cache

Table 6.3: Table of specifying attributes



## 6.4 Defining Operations

Operations define the behavior of an object. Operations are found by verb parsing from the story.

In Table 6.4 I have shown the table of defining operation of CAMATF

Noun	Attributes	Methods
User	+userId, +username, +email, +password, +numberOfPreviousTest +result	+initiateTestHistory() +visualizeResult() +saveResult() +viewPreviousTests()
Configuration	+display size +RAM +CPU +networkDelay +packetLoss +systemImage +configurationFile +virteualDevice	+createConfigFile() +instantiateVirtualDevice() +getter() +setter()
EmulatorManager	+vertualDevice +testCase +logFile +result	+createVirtualDevice() +runTest() +monitorVirtualDevice() +resultProcessing()
System	+username +email +password +userId	+createAccount() +validateAccount()

Table 6.4: Table of defining operations

## 6.5 Class Relation Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

In figure 6.1 I have shown the class responsibility diagram of CAMATF

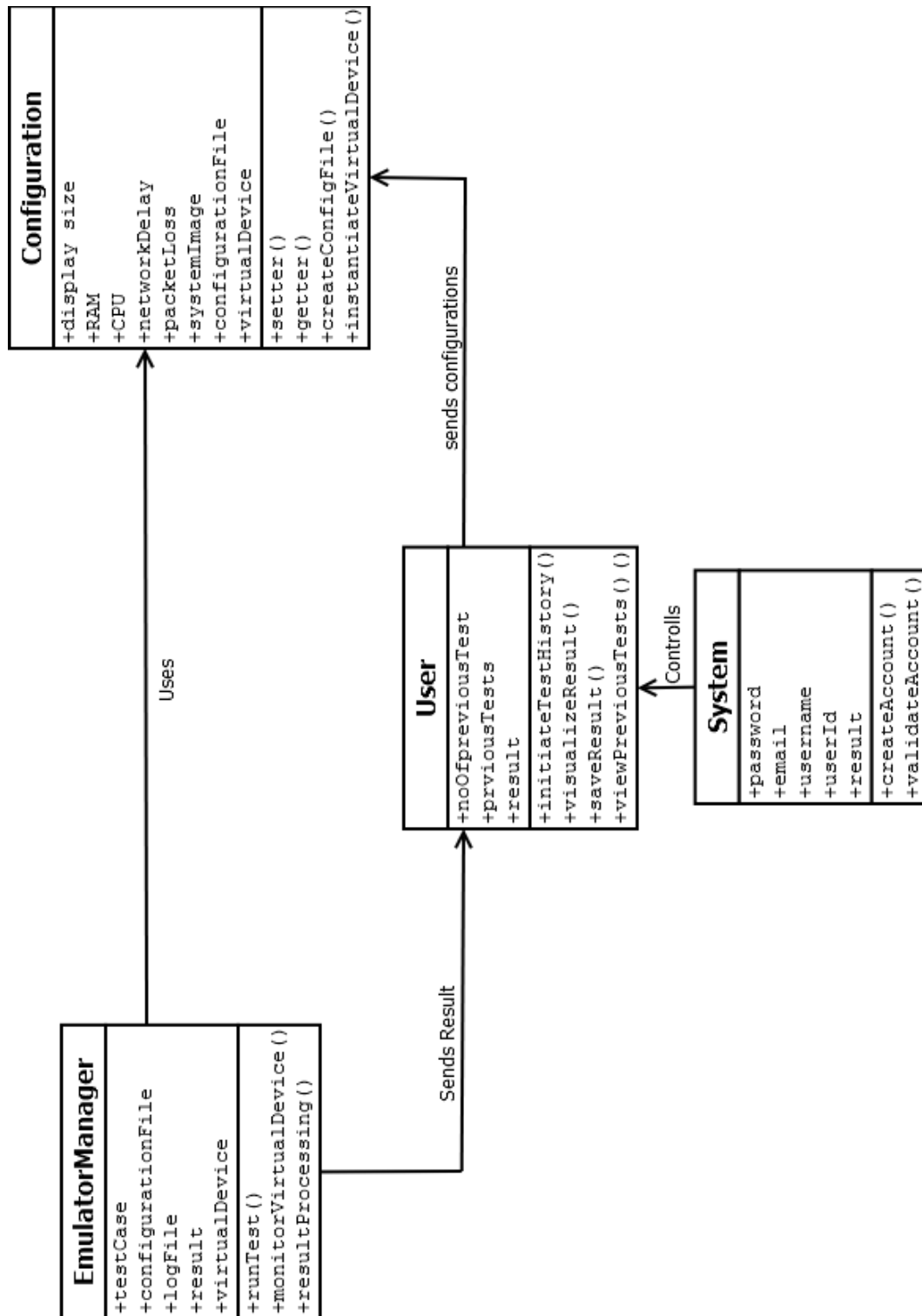


Figure 6.1: Figure of class relation diagram

## 6.6 Class-Responsibility-Collaborator (CRC) Model

A CRC model is really a collection of standard index cards that represent classes. The cards are divided into three sections. Along the top of the card you write the name of the class. In the body of the card you list the class responsibilities on the left and the collaborators on the right.

<b>Class: User</b>	
<b>Responsibilities</b>	<b>Collaborators</b>
Showing previous tests	User
Saving result	User, System
Visualizing Result	User, Emulator Manager

<b>Class: Configuration</b>	
<b>Responsibilities</b>	<b>Collaborators</b>
Collecting Contexts	User
Creating configuration file of virtual device	Emulator Manager, System
Instantiating Virtual Device	Emulator manager

<b>Class: EmulatorManager</b>	
<b>Responsibilities</b>	<b>Collaborators</b>
Running test	Emulator Manager
Processing test result	Emulator Manager
Monitoring virtual device	Emulator Manager

<b>Class: System</b>	
<b>Responsibilities</b>	<b>Collaborators</b>
Creating account	User
Validating Account	User

## Chapter 7: Behavioral Modeling

This modeling notation represents static elements of the requirements model. It describes the dynamic behavior of the system or product. Behavior modeling represents the behavior of the system as a function of specific events and time. Behavioral model indicates how software will respond to external events or stimuli.

### 7.1 Identifying Events

In Table 7.1 I have shown the table of event identification of CAMATF

<b>Event</b>	<b>Initiator</b>	<b>Collaborator</b>
Login	User	System
Create Account	User	System, User
Run Test	User	User, Configuration, EmulatorManager
View previous test result	User	User

Table 7.1: Event identification table

## 7.2 State Diagram

State diagram represents active states for each class and the events (triggers) that cause changes between these active states.

In figure 7.1 I have shown the state diagram of user class

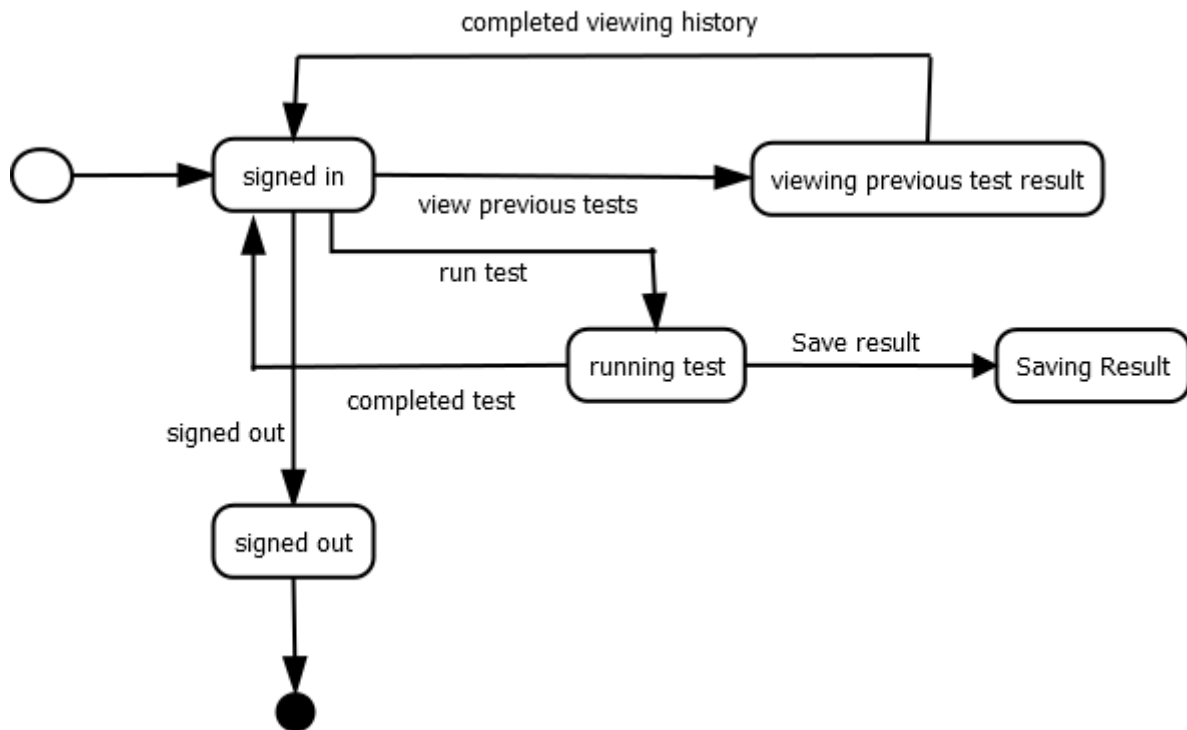


Figure 7.1: State Diagram of User class

In figure 7.2 I have shown the state diagram of configuration class

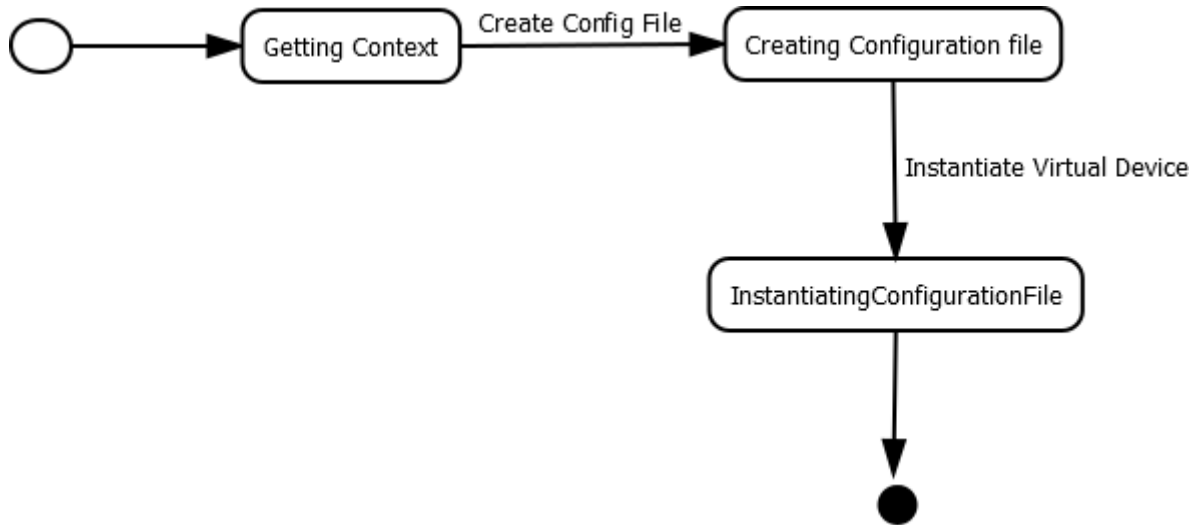


Figure 7.2: State Diagram of Configuration class

In figure 7.3 I have shown the state diagram of emulator manager class

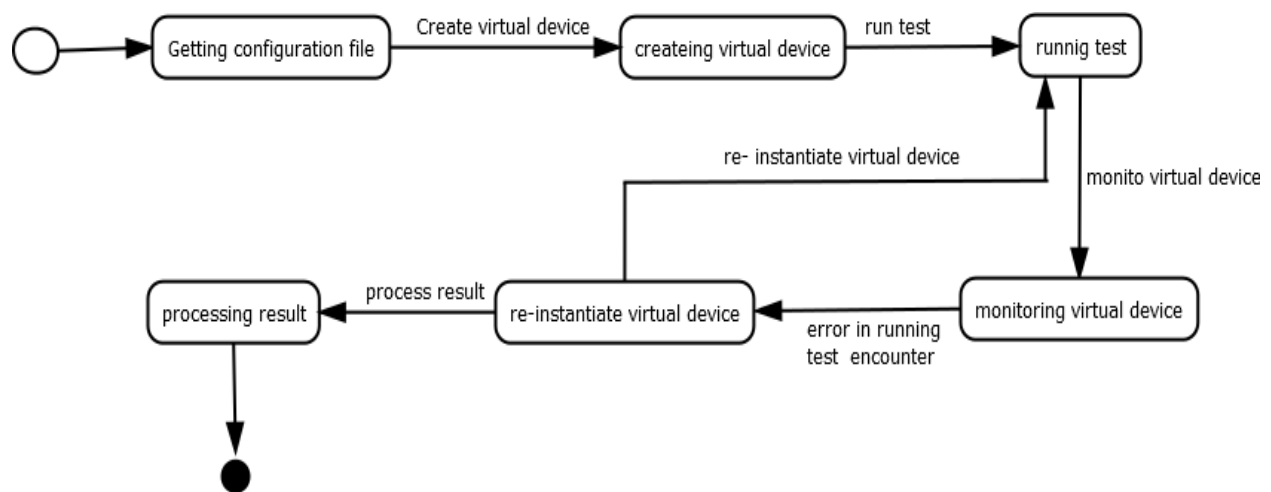


Figure 7.3: State Diagram of Emulator Manager Class

In figure 7.4 I have shown the state diagram of system class

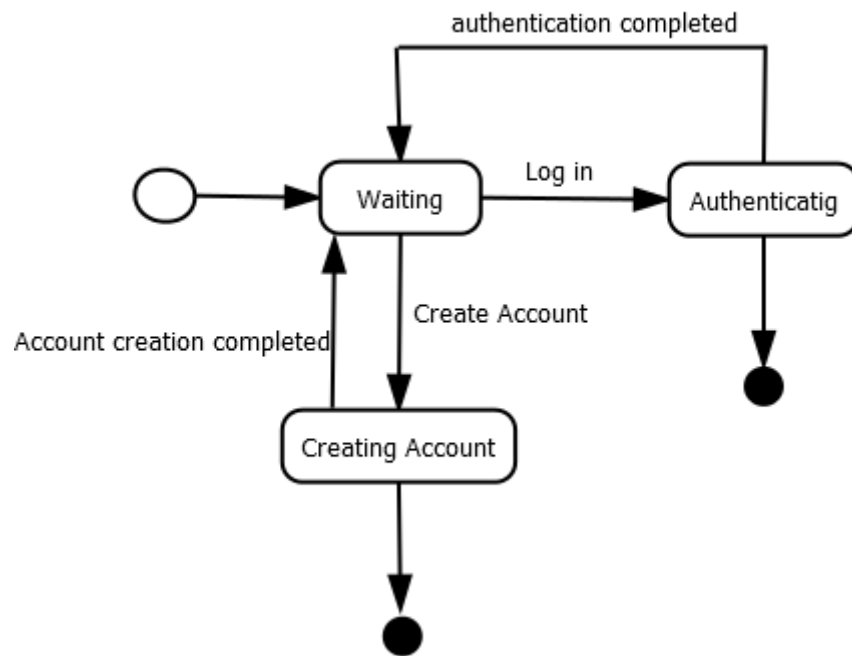


Figure 7.4: State Diagram of System Class

## 7.3 Sequence Diagram

Sequence diagram indicates how events cause transitions from object to object. Once events have been identified by examining a use case, the modeler creates a sequence diagram a representation of how events cause flow from one object to another as a function of time. In essence, the sequence diagram is a shorthand version of the use case. It represents key classes and the events that cause behavior to flow from class to class.

In figure 7.5 I have shown the sequence diagram of log in event

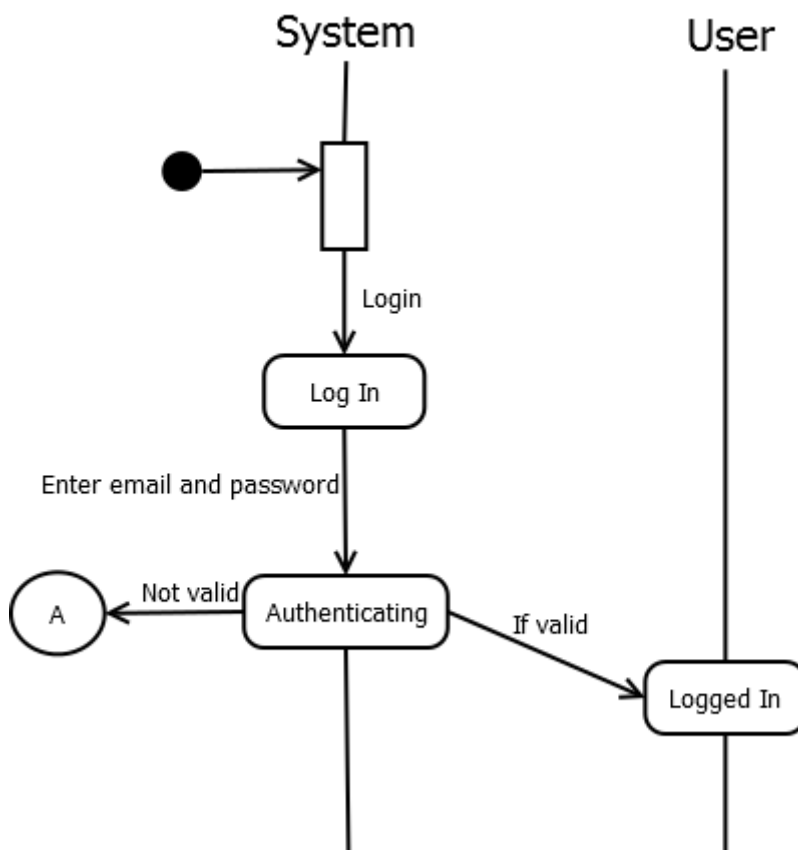


Figure 7.5: Sequence Diagram for log in event



In figure 7.6 I have shown the sequence diagram of create account event

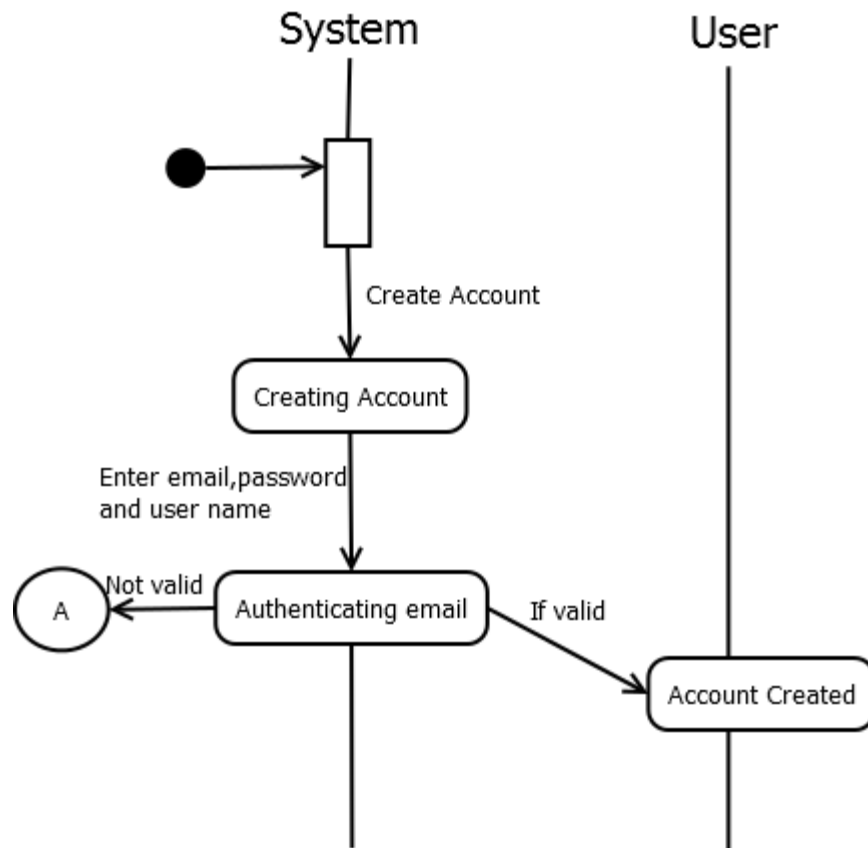


Figure 7.6: Sequence Diagram for create account event

In figure 7.7 I have shown the sequence diagram of run test event

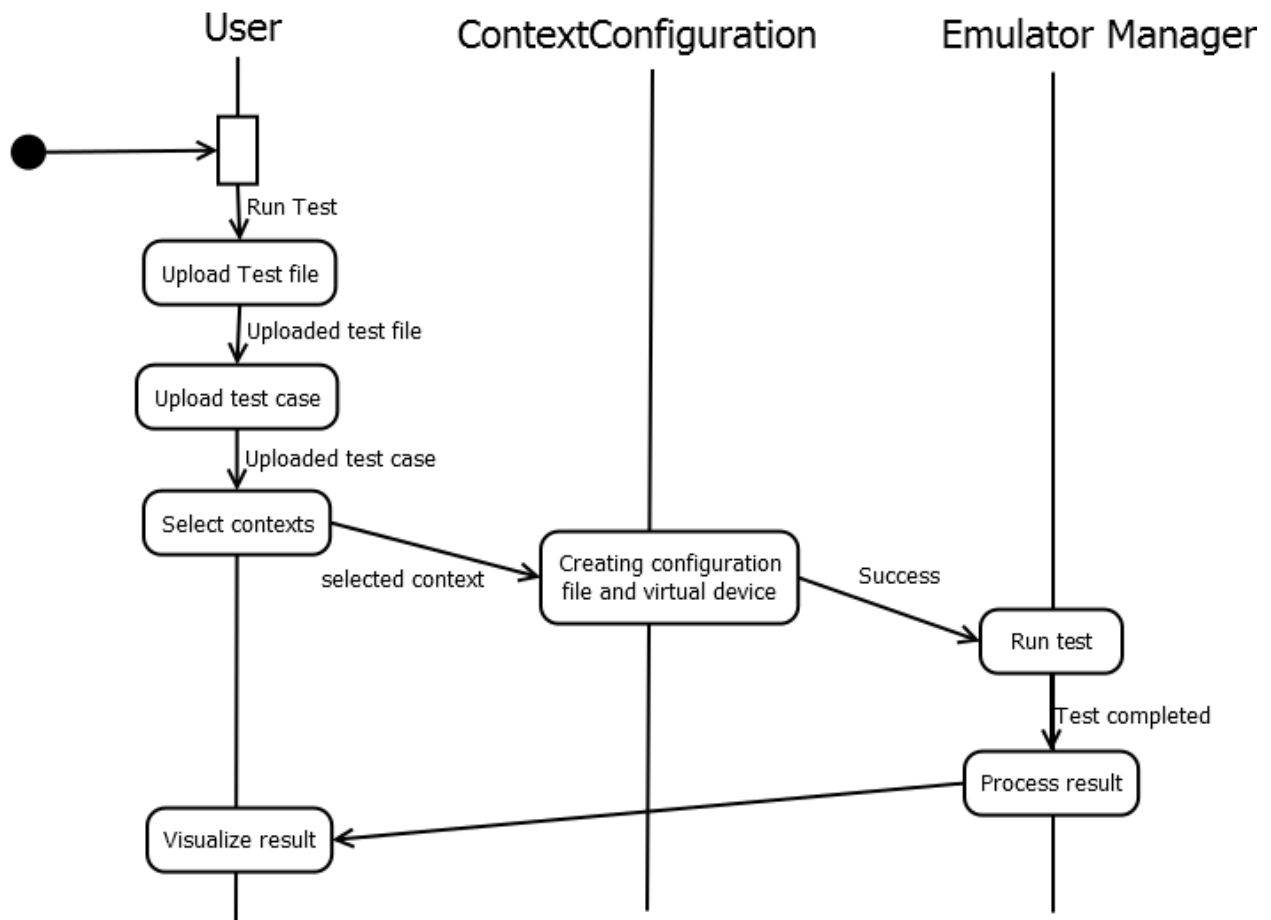


Figure 7.7: Sequence Diagram for run test event

In figure 7.8 I have shown the sequence diagram of view previous test event

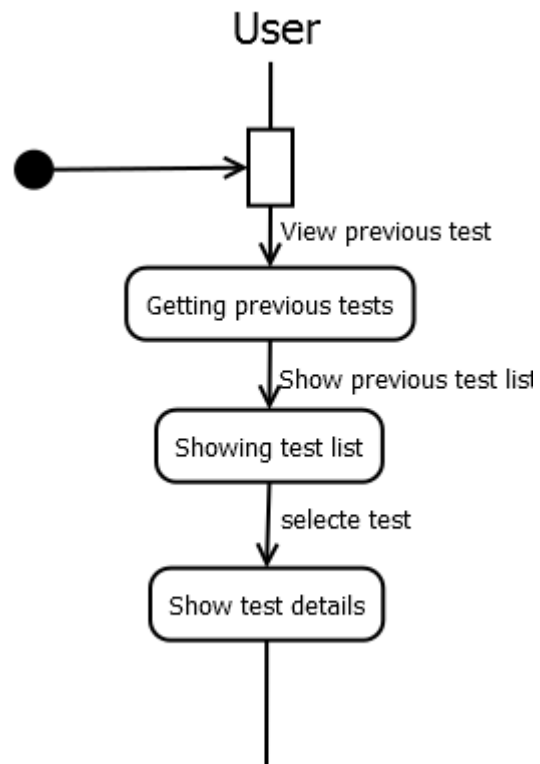


Figure 7.8: Sequence Diagram for view previous test

## Chapter 8: Conclusion

This document is written keeping in my mind the developers and the stakeholders of the product. So, there is a lot of technical term involved in this document. Though, we have tried to explain any complex technical term, but were not able to remove them all. We apologize for any prospect of this document that leads to failure of understanding. We also apologize for any inconvenience in the document.

## Chapter 9: References

1. Smart Bear , URL: <http://blog.smartbear.com/test-automation/the-demand-for-mobile-test-automation/> (last accessed on 8/11/2016)
2. Software Engineering A Practitioner's Approach